Problem A. BBB

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Byteasar has an account at The Byteotian Bit Bank (BBB in short). At the beginning there were p and at the end q bythalers in the account. Each transaction was either a deposit or a withdrawal of one bythaler. The account's balance was never negative. A bank teller has prepared a bank statement: a strip of paper with a sequence of pluses and minuses in it (a plus denotes a deposit while minus a withdrawal of one bythaler). Soon it turned out, that some transactions were not entered correctly. The bank teller cannot print another statement, but has to correct the one already printed instead. The statement needs not be consistent with the truth, it will suffice if the sequence of transactions satisfies the following two conditions:

- the final balance is consistent with the initial balance and the sequence of transactions in the statement,
- according to the sequence of transactions in the statement, the account's balance was never negative.

You are to calculate the minimum amount of time the bank teller needs to correct the bank statement.

The bank teller can:

- in x seconds turn an arbitrarily chosen transaction to its opposite, or
- in y seconds remove the last transaction and put it at the beginning of the statement.

If, for example, $p = 2$, $q = 3$, then `--++-+-++-+-+` is a correct statement. On the other hand the statement `---++++++` is incorrect, because the account's balance would become negative after the third transaction, and furthermore the final balance should be 3, not 5. It can be, however, corrected by turning the second to last symbol to its opposite and placing the last transaction at the beginning of the statement.

Write a programme that:

- reads the current bank statement for Byteasar's account (a sequence of pluses and minuses) as well as the numbers p, q, x and y from the standard input.
- writes out to the standard output the minimum number of seconds needed to correct the statement in a way such that the initial and final balance are consistent and that the balance is never negative.

Input

The first line contains 5 integers n, p, q, x and y ($1 \le n \le 1\,000\,000$, $0 \le p, q \le 1\,000\,000$, $1 \le x, y \le 1000$), separated by single spaces and denoting respectively: the number of transactions done by Byteasar, initial and final account balance and the number of seconds needed to perform a single turn (change of sign) and move of transaction to the beginning. The second line contains a sequence of n signs (each a plus or a minus), with no spaces in-between.

Output

The first and only output line should contain one integer, the minimum number of seconds needed to correct the statement. If no corrections are necessary, the number is zero. You may assume that a proper sequence of modifications exists for each test data.

Example

standard input	standard output
9 2 3 2 1 ---+++++++	3

Problem B. Close

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 megabytes

A known Polish proverb is: 'The apple always falls close to the apple tree'[1]. Your task is to verify this proverb experimentally. Positions of apple trees and apples can be identified with points in a plane. Distance between points is defined by Manhattan metric:

$$d((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$$

Let us assume that every apple has fallen from some apple tree closest to it.

Write a programme that:

- reads a description of positions of apple trees and apples from the standard input,
- computes the smallest distance between an apple and its apple tree,
- writes the result to the standard output.

Input

The first line of input contains two integers n and m ($1 \le n, m \le 100\,000$), separated with a single space and denoting the number of apple trees and the number of apples. The second line of input contains $2 \cdot n$ integers from the range $[0, 10^8]$, separated with single spaces and denoting the pairs of coordinates of apple trees: $x_1\ y_1\ x_2\ y_2\ \ldots\ x_n\ y_n$. The third line of input contains $2 \cdot m$ integers from the range $[0, 10^8]$, separated with single spaces and denoting the pairs of coordinates of apples: $x'_1\ y'_1\ x'_2\ y'_2\ \ldots\ x'_m\ y'_m$. We treat both apple trees and apples as points in a plane; there can be multiple apple trees and multiple apples in one point.

Output

The first and only line of output should contain the smallest distance between an apple and its apple tree.

Example

standard input	standard output
3 4 0 2 2 0 2 2 1 1 2 1 2 3 1 4	1

[1] An English equivalent of this proverb is: 'Like father, like son' or 'Like mother, like daughter'.

Problem C. The Great Escape

Input file:	`standard input`
Output file:	`standard output`
Time limit:	20 seconds
Memory limit:	64 megabytes

Al Bytone, the infamous thief, is planning a bank robbery. He knows only too well that the moment he robs the bank a pursuit will be commenced. Unfortunately, Al Bytone is a poor driver and turning left causes him great trouble. This is why he tries to devise such an escape route that at each intersection he would either ride straight ahead or turn right. He is also aware that once he passes through any intersection, the police will come and remain there, waiting for him. Therefore he may pass through any intersection at most once. Furthermore, the police are always present at certain intersections, so Al Bytone will have to avoid these intersections as well (there's no police at the intersections near the bank and near Al Bytone's hideout.)

Al Bytone is planning his escape route. To your great (and rather unpleasant) surprise, he paid you a visit and told to calculate the number of different escape routes leading from the bank to his hideout complying with the aforementioned requirements. Needless to say, Al Bytone does not take 'no' as an answer...

The streets of Byteburg form a rectangular grid. Every street runs either in the North-South or East-West direction, and every two non-parallel streets intersect. The bank is situated to the south of the south-western-most intersection. Al Bytone will start his great escape driving north.

Write a programme that:

- reads from the standard input the location of hideout, descriptions of intersections with police and a positive integer k,
- calculates the number of different escape routes leading from the bank to the hideout complying with the aforementioned requirements,
- writes out to the standard output the result modulo k.

Input

There are three integers in the first line of the standard input, n, m and k ($1 \le n, m \le 100$, $1 \le k \le 10^9$). The numbers n and m denote the number of streets leading in East-West and North-South direction, respectively. The second line contains two integers x and y ($1 \le x \le m$, $1 \le y \le n$). These represent the hideout's location — at the intersection of x^{th} street leading in North-South direction and y^{th} street leading in East-West direction. The streets are numbered from West to East and from North to South, from 1 to m and from 1 to n, respectively.

Each of the following n lines contains m characters „`*`" and/or „`+`". This is the city map. The character in i^{th} line and j^{th} column of the map depicts the intersection of the i^{th} street leading from West to East with the j^{th} street leading from North to South. „`*`" means there is police at the intersection, while „`+`" means there is no police there, so the escape route may pass through this intersection.

Al Bytone starts his great escape driving onto the intersection with coordinates $(1, n)$ from the South, i.e. from a nonexistent intersection $(1, n + 1)$.

Output

Your programme should write out the remainder of the number of escape routes modulo k in the first and only line of the standard output.

Example

standard input	standard output
3 5 10 4 2 +++++ ++*++ ++++*	2

Problem D. Lights

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	256 megabytes

Little Johny has received a most unusual Christmas present. The sign on the freshly unwrapped box read "Infinite chain of fairy lights". Amused, the boy laid out his new toy on the floor.

Johny's chain is a cable with but one end: it begins at some point, but does not end anywhere. Attached to the cable are fairy lights, numbered (in order of attachment) with successive natural numbers, starting with 0. The cable itself is plugged to a control panel. There is a number of buttons on the panel, each uniquely colored and each inscribed with unique positive integer. The numbers inscribed to the buttons are pairwise relatively prime.

Upon unwrapping, no light was turned on. Thinking little at the time, Johny pressed all the buttons one by one, from first to last. To his increasing amusement he noticed that pressing the i^{th} button turns on exactly the lights which numbers are multiples of p_i, the number inscribed on the i^{th} button. Moreover, they are burning in the color k_i, the one of the button. In particular, all the lights whose numbers are multiples of p_i that were previously lit, change their color to k_i.

Johny gazes infatuated at the infinite multicolour chain and wonders what fraction of the lights burns with each particular colour. Let $L_{i,r}$ denote the number of lights burning with the colour k_i among the lights with numbers $0, 1, \ldots, r$. Formally, the fraction C_i of the lights burning with colour k_i is defined as:

$$C_i = \lim_{r \to \infty} \frac{L_{i,r}}{r}$$

Write a programme that:

- reads the descriptions of the buttons on the control panel from the standard input,
- for each colour k_i calculates the fraction C_i, denoting the fraction of lights burning with the colour k_i,
- writes out the result to the standard output.

Input

The first line of the standard input contains a single integer n ($1 \le n \le 1\,000$), denoting the number of buttons on the control panel. Each of the following n lines contains a single integer p_i ($1 \le p_i \le 1\,000\,000\,000$), meaning that pressing the i^{th} button makes the lights numbered with multiples of p_i burn with the colour k_i. The numbers p_i are given precisely in the order Johny had pressed them. The numbers p_i are pairwise relatively prime (and thus different).

Output

Your programme should write out exactly n lines to the standard output. The i^{th} line should contain the fraction C_i of the lights burning with colour k_i, written as a fraction a/b, where a is an integer, b is a positive integer and a and b are relatively prime. If $C_i = 0$, the fraction should be written as $0/1$.

Example

standard input	standard output
3 2 3 5	4/15 4/15 1/5

Problem E. Permutation (High Div Only)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	256 megabytes

Multiset is a mathematical object similar to a set, but each member of a multiset may occur in it more than once. Just as with any set, the members of a multiset can be ordered in many ways. We call each such ordering a *permutation* of the multiset. For example, among the permutations of the multiset $\{1, 1, 2, 3, 3, 3, 7, 8\}$ there are $(2, 3, 1, 3, 3, 7, 1, 8)$ and $(8, 7, 3, 3, 3, 2, 1, 1)$.

We will say that one permutation of a given multiset is smaller (in lexicographic order) than another permutation, if on the first position that does not match the first permutation has a smaller element than the other one. All permutations of a given multiset can be numbered (starting from one) in an increasing order.

Write a programme that

- reads the description of a permutation of a multiset and a positive integer m from the standard input,
- determines the remainder of the rank of that permutation in the lexicographic ordering modulo m,
- writes out the result to the standard output.

Input

The first line of the standard input holds two integers n and m ($1 \le n \le 300\,000$, $2 \le m \le 1\,000\,000\,000$), separated by a single space. These denote, respectively, the cardinality of the multiset and ... the number m. The second line of the standard input contains n positive integers a_i ($1 \le a_i \le 300\,000$), separated by single spaces and denoting successive elements of the multiset permutation.

Output

The first and only line of the standard output is to hold one integer, the remainder modulo m of the rank of the input permutation in the lexicographic ordering.

Example

standard input	standard output
4 1000 2 1 10 2	5

All the permutations smaller (with respect to lexicographic order) than the one given are: $(1, 2, 2, 10)$, $(1, 2, 10, 2)$, $(1, 10, 2, 2)$ and $(2, 1, 2, 10)$.

Problem F. Postering

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

All the buildings in the east district of Byteburg were built in accordance with the old arbitecture: they stand next to each other with no spacing inbetween. Together they form a very long chain of buildings of diverse height, extending from east to west.

The mayor of Byteburg, Byteasar, has decided to have the north face of the chain covered with posters. Byteasar ponders over the minimum number of posters sufficient to cover the whole north face. The posters have rectangular shape with vertical and horizontal sides. They cannot overlap, but may have common sides. Every poster has to entirely adjoin the walls of certain buildings and the whole surface of the north face has to be covered.

Write a programme that:

- reads the description of buildings from the standard input,
- determines the minimum number of posters needed to entirely cover their north faces,
- writes out the outcome to the standard output.

Input

The first line of the standard input contains one integer n ($1 \leq n \leq 250\,000$), denoting the number of buildings the chain comprises of. Each of the following n lines contains two integers d_i and w_i ($1 \leq d_i, w_i \leq 1\,000\,000\,000$), separated by a single space, denoting respectively the length and height of the i^{th} building in the row.

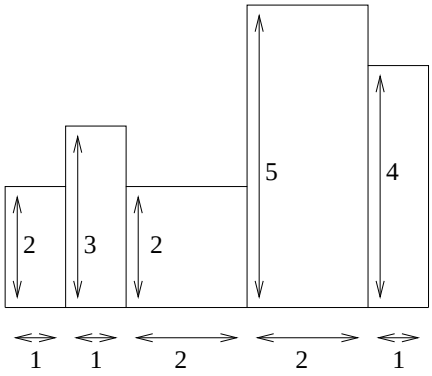
Output

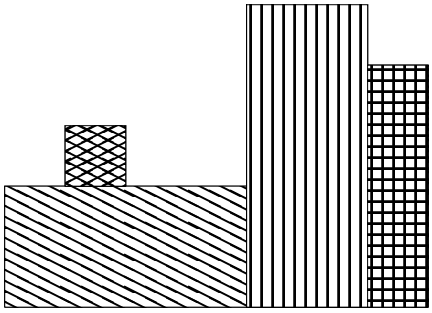
The first and only line of the standard output should contain one integer, the minimum number of rectangular posters that suffice to cover the north faces of the buildings.

Example

standard input	standard output
5 1 2 1 3 2 2 2 5 1 4	4

The following figures show the north face of the buildings chain. The second figure shows an exemplary covering of the face with four posters.


2
3
2
5
4
1
1
2
2
1



Problem G. Robinson (High Div Only)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	512 megabytes

Tossed by the storm on a deserted island, Robinson built himself a boat so that he could go out to the sea and seek out human domicile. He is an experienced sailor, therefore he built the boat with accordance to the rules of craftsmanship: it has a longitudinal axis of symmetry and an appropriate shape. The boat's prow is thin, and it widens gradually towards the boat's centre, only to gradually narrow once again towards the stern. In particular, at some point in the middle the boat is wider than both at the prow and stern.

Unfortunately, Robinson has launched his boat in a most improper space: there is extremely thick reed all around. It is, moreover, so stiff that the boat cannot break through. Perhaps Robinson can get to the high seas by carefully manoeuvring between the reed.

Due to lack of manoeuvrability, the boat can move forward and backward and even sidewards (leftward or rightward), but it cannot turn. It is thus allowed, and may be in fact necessary, that the boat moves with its stern or broadside to the front.

You are to verify whether Robinson can get to the high seas.

To make your task easier the island and its surroundings will be represented by a square map divided into square unit fields, each occupied by either water, part of Robinson's boat or an obstacle, e.g. land or reed. Initially the boat is set parallel to one of the cardinal directions, i.e. its longitudinal axis of symmetry is parallel to this direction and the axis bisects the unit fields it is covered with.

We assume that the high seas starts where the map ends. Hence Robinson may get to the high seas if his boat can leave the area depicted in the map. A single move consists in moving the boat to a side-adjacent field in a chosen direction (north, south, east or west). The move is permissible if both before and after it the boat remains entirely in water (it does not occupy a field with an obstacle).

You are to write a programme that

- reads the map's description from the standard input,
- calculates the minimum possible number of boat's moves that suffice to completely leave the area depicted in the map,
- writes out this number to the standard output.

Input

The first line contains one integer $3 \leq n \leq 2000$, denoting the length of the map's side. In each of the following n lines there are n characters describing successive fields of the map: i^{th} character in the $(j+1)^{th}$ line tells the contents of the field (i, j). The following characters may appear there:

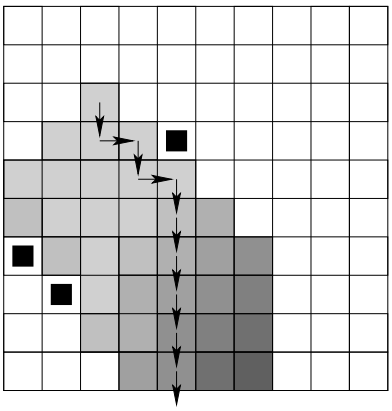
- `.` — (dot) denotes a field filled with water,
- `X` — denotes an obstacle (land or reed),
- `r` — denotes a part of Robinson's boat.

Output

Your programme should write out (in the first and only line of the standard output) a single positive integer, equal to the minimum number of boat's moves that suffice to completely leave the area depicted in the map. Should getting to the high seas be impossible, write out the word 'NIE' ('no' in Polish).

Example

standard input	standard output
10 ....r....... .rrrX..... rrrrr..... .rrr...... X.r....... .Xr.......	10



Problem H. Toll

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

King Byteasar has yielded under pressure of Byteotian merchants and hence decided to settle the issue of toll paid by them. Byteotia consists of n towns connected with m bidirectional roads. Each road connects directly two different towns and no two towns are connected by more than one direct road. Note that the roads may lead through tunnels or flyovers.

Until now each town in Byteotia imposed duty on everyone who either entered or left the town. The merchants, discontented with such situation, lodged a protest against multiple taxation. King Byteasar ruled that the town privileges are now restricted. According to the new royal edict, each town can only charge toll on merchants travelling by exact one road leading into the town, regardless of the direction they are travelling in. Furthermore, for each road, those who travel it cannot be made to pay the duty to both towns the road connects. It remains to determine which town should collect toll from which road. Solving this problem His Highness has commissioned to you.

Write a programme that:

- reads the Byteotian road system's description from the standard input,
- for each town determines on which road it should impose toll, or claims it is impossible,
- writes out the result to the standard output.

Input

There are two integers in the first line of the standard input: n and m ($1 \leq n \leq 100\,000$, $1 \leq m \leq 200\,000$), denoting the number of towns and roads in Byteotia, respectively. The towns are numbered from 1 to n. In next m lines descriptions of the roads follow. In line no. i there are two integers a_i and b_i ($1 \leq a_i < b_i \leq n$) meaning the towns a_i and b_i are directly connected by a road.

Output

If collecting the toll in accordance with the royal edict is impossible, your programme should write the word `NIE` ('no' in Polish) in the first and only line of the standard output. Otherwise, it should write the word `TAK` ('yes' in Polish) in the first line, while in the following n lines should tell which city collects toll from which road. Line no. $i+1$ should tell on which road the town no. i imposes toll. Since town no. i is obviously one endpoint of the road, it is enough to tell what is the other endpoint. Thus if the town no. i imposes toll on the road connecting it with the town no. j, the line no. $i+1$ should contain the number j. If more than one solution exists, write out one chosen arbitrarily.

Example

standard input	standard output
4 5 1 2 2 3 1 3 3 4 1 4	TAK 3 3 4 1

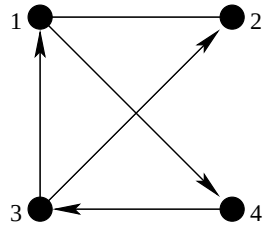


Illustration of the first example: The arrows in the figure point to towns that collect toll from merchants using the road. Observe that the merchants who follow the road connecting the towns 1 and 2 are not charged with duty at all.

standard input	standard output
4 3 1 3 3 4 2 3	NIE

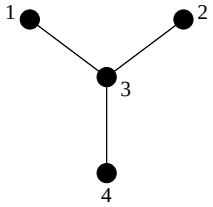


Illustration of the second example.

Problem I. Triangles

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	256 megabytes

n pairwise disjoint points in the plane are given ($n \geq 3$). There are $\frac{n \cdot (n-1) \cdot (n-2)}{6}$ triangles whose vertices are some pairwise different points among them (including degenerate triangles, i.e. ones whose vertices are collinear).

We wish to calculate the sum of areas of all the triangles with vertices in the given points.

Those parts of the plane that belong to many triangles are to be calculated multiple times. We assume that the area of degenerate triangles (i.e. those with collinear vertices) is zero.

Write a programme that:

- reads from the standard input the coordinates of the points in the plane,
- determines the sum of the areas of all the triangles with vertices in the given points,
- writes out the result to the standard output.

Input

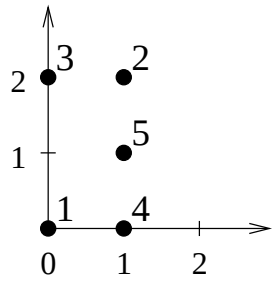
In the first line of the standard input there is one integer n ($3 \leq n \leq 3\,000$) denoting the number of selected points. Each of the following n lines contains two integers x_i and y_i ($0 \leq x_i, y_i \leq 10\,000$) separated by a single space and denoting the coordinates of the i^{th} point (for $i = 1, 2, \ldots, n$). No pair (ordered) of coordinates appears more than once.

Output

In the first and only line of the standard output there should be one real number equal to the sum of the areas of all the triangles with vertices in the given points. The outcome should be printed out with exactly one digit after dot and should not differ from the correct value by more than 0.05.

Example

standard input	standard output
5 0 0 1 2 0 2 1 0 1 1	7.0



Problem J. Turns

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 megabytes

Byteman decided to go for a car ride to the mountains. Although he has a map, he is not able to locate himself on the map.

The road on the map is represented by n turns, each of which is 90° turn to the right or to the left. For simplicity we can represent map by n-letters string consisting of `L` and/or `P` letters.

Assuming that Byteman has the i'th turn in front of himself (however he might not know that it is the i^{th} turn), by a_i we denote the number of turns Byteman has to drive through, in order to be sure in which place on the map he is located.

Let us explain the meaning of the number a_i using an example. Let us assume that the road on the map is represented by a string `LLPPLPL`. If Byteman is in front of his second turn, before passing that turn he knows that he could be in front of one of the turns: 1, 2, 5 or 7 (as he sees a left turn in front of him). After passing that turn, Byteman sees a turn to the right — `P` — what means that initially he could not had been in front of the first turn (as the following turn was to the left) as well as the last turn (as it is followed by the end of the road). Therefore he knows that he is in front of third or sixth turn on the map. After passing that turn, Byteman sees a turn to the right — `P` — what means that initially he could not had been in front of the fifth turn. This leads to the observation that after passing two turns Byteman is in front of the fourth turn, so $a_2 = 2$.

Write a programme that

- reads from the standard input a description of the map,
- determines values a_i, for $1 \le i \le n$,
- writes out the result to the standard output.

Input

The first line of the standard input contains one integer n ($1 \le n \le 500\,000$). The second line contains a description of the road — n letters `L` and/or `P`, without any spaces.

Output

Your programme should write out exactly n lines. The i^{th} line should contain one integer a_i.

Example

standard input	standard output
7 LLPPLPL	1 2 1 2 2 2 1

Problem K. n-Dimensional Farm (Div 1 only!)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

f course you know that n-dimensional Kittens live in the n-dimensional space and build true n-dimensional towns. In the town named Transborough their houses are located only in points with integer coordinates in range from 1 to n. When they planned to build this town, they had decided that all the points with at least two equal coordinates are bad for building a house, so now only points with different coordinates are occupied and there are no more points free for building a house now because someone lives at any good point.

Recently the local authorities have built a milk farm near the town. All Kittens visit this farm every day because they like milk very much. Of course any Kitten spends some time walking from home to the farm and back.

The Kittens are arguing who's the luckiest Kitten who lives at the minimal distance to the farm. Help them to establish the truth!

Input

The first line of the input file contains one integer number n ($1 \le n \le 64$). The second line is filled with n numbers separated by spaces. These numbers are coordinates of the milk farm. All coordinates are integers in range from -10000 to 10000.

Output

Write to the output file exactly n numbers separated by a single space — the coordinates of the Kitten's house nearest to the milk farm. If there is a tie, output any of the optimal answers.

Example

standard input	standard output
2 0 1	1 2

Problem L. First Occurence (Div 1 Only!)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

asya is constructing a sequence consisting of digits 0, 1 and 2. He's written a single digit 0 and now he repeats the following steps: he takes the written sequence and copies it two times more, each time replacing $0 \to 1$, $1 \to 2$, $2 \to 0$ so from a sequence of n digits he obtains a $3n$-digit one. For example, the first two his steps are $0 \to 012 \to 012120201$. His goal is to find the first occurence of substring S in this sequence.

Write a program that will find the first occurence of substring S in the sequence being constructed to let Vasya have some sleep...

Input

The input file contains only one line consisting of digits 0, 1 and 2. The length of the line does not exceed 255 characters. The input line cannot be empty.

Output

Write to the output file the first position in this sequence where the substring S occurs. If the substring S does not occur in the Vasya's sequence, output -1.

Example

standard input	standard output
12	2